

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ**

Ордена Трудового Красного Знамени

**Федеральное государственное бюджетное образовательное
учреждение высшего образования**

«Московский технический университет связи и информатики»

Лабораторная работа № 3

«Создание хэш-таблиц»

Выполнила: Студентка группы БВТ2306

Максимова Мария

Москва 2024

Гитхаб: <https://github.com/Mashaaaaa7/itip-3>

Задачи:

Задание 1:

1. Создайте класс HashTable, который будет реализовывать хэш-таблицу с помощью метода цепочек.
2. Реализуйте методы put(key, value), get(key) и remove(key), которые добавляют, получают и удаляют пары «ключ-значение» соответственно.
3. Добавьте методы size() и isEmpty(), которые возвращают количество элементов в таблице и проверяют, пуста ли она

Задание 2: Работа с встроенным классом HashMap.

Вариант 4: Реализация хэш-таблицы для хранения информации о книгах в библиотеке. Ключом будет ISBN книги, а значением - объект класса Book, содержащий информацию о названии, авторе и количестве копий. Необходимо реализовать операции вставки, поиска и удаления книги по ISBN.

При выполнении задания 1 реализуем методы put, get, remove (key) и добавим методы Size и isEmpty.

Код к заданию 1:

```
import java.util.HashMap;

class KeyValue { //представляет пару ключ-значение
    private String key;
    private String value;

    public KeyValue(String key, String value) { //хранит ключ и значение
        this.key = key;
        this.value = value;
    }

    public String getKey() {
        return key;
    }

    public String getValue() {
        return value;
    }
}

public class HashTable {
```

```

    private HashMap<String, KeyValue> keyMap; //создает поле keyMap типа
    HashMap, которое будет хранить пары ключ-значение

    public HashTable() {
        keyMap = new HashMap<>(); //инициализирует keyMap пустым HashMap.
    }

    public void put(String key, KeyValue value) { //добавляет пару ключ-
    значение в таблицу.
        keyMap.put(key, value);
    }

    public KeyValue get(String key) { //возвращает объект KeyValue, связанный
    с указанным ключом. Не найден=null
        return keyMap.get(key);
    }

    public void remove(String key) {
        keyMap.remove(key);
    }

    public int size() { //количество
        return keyMap.size();
    }

    public boolean isEmpty() { //пустая таблица
        return keyMap.isEmpty();
    }

    /**В методе main создается объект HashTable, добавляются пары ключ-
    значение, выводится размер таблицы,
    // и выводится результат проверки на пустоту таблицы.

    public static void main(String[] args) {
        HashTable HashTable = new HashTable(); //создает объект HashTable
        HashTable.put("name", new KeyValue("name", "George Doe"));
        HashTable.put("age", new KeyValue("age", "21"));

        System.out.println("Размер таблицы: " + HashTable.size());

        KeyValue value = HashTable.get("name"); //получение значения по ключу
name
        if (value != null) { //проверяет, найдено ли значение.
            System.out.println("Значение для ключа 'name': " +
value.getValue());
        } else {
            System.out.println("Ключ 'name' не найден.");
        }

        value = HashTable.get("age"); //получение значения по ключу age
        if (value != null) { //проверяет, найдено ли значение
            System.out.println("Значение для ключа 'age': " +
value.getValue());
        } else {
            System.out.println("Ключ 'age' не найден.");
        }

        HashTable.remove("name"); //удаляет пару по ключу "name"
        HashTable.remove("age"); //удаляет пару по ключу "age"

        System.out.println("Размер таблицы: " + HashTable.size()); //
        (выводит размер таблицы после удаления)

        System.out.println("Пуста ли таблица: " + HashTable.isEmpty()); //

```

```
(пуста ли таблица)
    }
}
```

Вывод в консоли:

```
Размер таблицы: 2
Значение для ключа 'name': George Doe
Значение для ключа 'age': 21
Размер таблицы: 1
Пуста ли таблица: false

Process finished with exit code 0
```

Рисунок 1

Этот код представляет простую имитацию хэш-таблицы с использованием класса `HashTable`, используя `HashMap`.

1. Внутренний класс `KeyValue`:

```
class KeyValue {
    private String key;
    private String value; //значение

    public KeyValue(String key, String value) {
        this.key = key;
        this.value = value;
    }

    public String getKey() {
        return key;
    }

    public String getValue() {
        return value;
    }
}
```

Рисунок 1

Публичный класс `HashTable`.

```
public class HashTable {
    private HashMap<String, KeyValue> keyMap;

    public HashTable() {
        keyMap = new HashMap<>();
    }

    public void put(String key, KeyValue value) {
        keyMap.put(key, value);
    }

    public KeyValue get(String key) {
        return keyMap.get(key);
    }

    public void remove(String key) {
        keyMap.remove(key);
    }
}
```

Рисунок 2

```

public int size() { 2 usages
    return keyMap.size();
}

public boolean isEmpty() { 1 usage
    return keyMap.isEmpty();
}

```

Рисунок 3

- Класс KeyValue представляет пару ключ-значение.
- private String key;: хранит ключ типа String.
- private String value;: хранит значение типа String.
- Класс HashTable содержит HashMap, в котором ключом является строка, а значением - объект класса KeyValue.
- Метод put добавляет пару ключ-значение в таблицу.
- Метод get возвращает значение по заданному ключу.
- Метод remove удаляет значение по ключу.
- Метод size возвращает размер таблицы (количество элементов).
- Метод isEmpty проверяет, пуста ли таблица.

2. Класс HashTable:

- public class HashTable: объявляет класс HashTable.
- HashMap<String, KeyValue> — тип поля keyMap. Это хеш-таблица, которая хранит пары ключ-значение.
- HashTable(): конструктор, который инициализирует keyMap пустым HashMap.
- keyMap = new HashMap<>(); — эта строка инициализирует поле keyMap новым объектом HashMap, создавая пустую хеш-таблицу.

Метод main:

```
public static void main(String[] args) {
    Hashtable hashtable = new Hashtable(); //создает объект Hashtable
    hashtable.put("name", new KeyValue("name", "George Doe"));
    hashtable.put("age", new KeyValue("age", "21"));

    System.out.println("Размер таблицы: " + hashtable.size());
}
```

Рисунок 4

```
KeyValue value = hashtable.get("name"); //получение значения по ключу name
if (value != null) { //проверяет, найдено ли значение.
    System.out.println("Значение для ключа 'name': " + value.getValue());
} else {
    System.out.println("Ключ 'name' не найден.");
}

value = hashtable.get("age"); //получение значения по ключу age
if (value != null) { //проверяет, найдено ли значение
    System.out.println("Значение для ключа 'age': " + value.getValue());
} else {
    System.out.println("Ключ 'age' не найден.");
}
```

Рисунок 5

В методе main создается объект Hashtable, добавляются пары ключ-значение, выводится размер таблицы, удаляется значение по ключу "name", удаляется значение по ключу "age" и выводится результат проверки на пустоту таблицы.

- `public static void main(String[] args):` точка входа в программу.
- `Hashtable hashtable = new Hashtable();`: создает объект Hashtable.
- `hashtable.put("name", new KeyValue("name", "John Doe"));`: добавляет пару “name” - “John Doe” в таблицу.
- `hashtable.put("age", new KeyValue("age", "30"));`: добавляет пару “age” - “21” в таблицу.
- `System.out.println("Размер таблицы: " + hashtable.size());`: выводит размер таблицы.
- `if (value != null) { ... }:` проверяет, найдено ли значение. Если да, то выводит его, иначе выводит сообщение о том, что ключ не найден.
- `hashtable.remove("age");`: удаляет пару по ключу “age”.
- `hashtable.remove("name");`: удаляет пару по ключу “name”.

- `System.out.println("Размер таблицы: " + hashTable.size());`: выводит размер таблицы после удаления.
- `System.out.println("Пуста ли таблица: " + hashTable.isEmpty());`: выводит, пуста ли таблица.

Далее, перейдем к заданию 2.

Мой код:

```
import java.util.HashMap;

class Book { //определяет класс, который представляет книгу
    private String title;
    private String author;
    private int copies;

    public Book(String title, String author, int copies) { //инициализирует
        //объект Book
        this.title = title;
        this.author = author;
        this.copies = copies;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    public int getCopies() {
        return copies;
    }

    // Метод для изменения количества копий
    public void setCopies(int copies) {
        this.copies = copies;
    }
}

public class Library { //определяет класс Библиотека
    private HashMap<String, Book> bookMap; // Явное объявление HashMap

    public Library() {
        bookMap = new HashMap<>(); //создает поле bookMap типа HashMap,
        //которое будет хранить книги с ключом ISBN.
    }

    public void addBook(String isbn, Book book) {
        bookMap.put(isbn, book); // Добавление книги в хэш-таблиц
    }

    public Book findBook(String isbn) {
        return bookMap.get(isbn);
    }

    public void removeBook(String isbn) {
        bookMap.remove(isbn);
    }

    public static void main(String[] args) {
        Library library = new Library(); //создает объект Library

        // Добавление книг
```



```

        library.addBook("978-0143034231", new Book("1984", "George Orwell",
5));
        library.addBook("978-0141439501", new Book("Pride and Prejudice",
"Jane Austen", 3));

        // Поиск книги
        Book foundBook = library.findBook("978-0143034231");
        if (foundBook != null) { //проверяет, найдена ли книга.
            System.out.println("Найдена книга: " + foundBook.getTitle() + "
by " + foundBook.getAuthor());
        } else {
            System.out.println("Книга не найдена.");
        }

        Book foundBook2 = library.findBook("978-0141439501");
        if (foundBook2 != null) {
            System.out.println("Найдена книга: " + foundBook2.getTitle() + "
by " + foundBook2.getAuthor());
        } else {
            System.out.println("Книга не найдена.");
        }

        // Удаление книги
        library.removeBook("978-0141439501");
        library.removeBook("978-0141439501");

        // Поиск удаленной книги
        foundBook = library.findBook("978-0141439501");
        if (foundBook != null) {
            System.out.println("Книга найдена.");
        } else {
            System.out.println("Книга не найдена.");
        }

        foundBook2 = library.findBook("978-0141439501");
        if (foundBook2 != null) {
            System.out.println("Книга найдена.");
        } else {
            System.out.println("Книга не найдена.");
        }
    }
}

```

Этот код представляет собой пример использования HashMap в Java для хранения книг в библиотеке.

В методе main создается объект класса Library и добавляются две книги. Затем находится книга по ISBN, выводится информация о ней. Книга удаляется по ISBN и затем снова производится поиск этой книги, результат которого выводится в консоль.

Класс Book:

```

class Book { //определяет класс, который представляет книгу 6 usages
    private String title; 2 usages
    private String author; 2 usages
    private int copies; 3 usages

    public Book(String title, String author, int copies) { //инициализирует объект Book с указанными названием, автором и количеством копий
        this.title = title;
        this.author = author;
        this.copies = copies;
    }

    public String getTitle() { 1 usage
        return title;
    }

    public String getAuthor() { 1 usage
        return author;
    }

    public int getCopies() { no usages
        return copies;
    }

    // Метод для изменения количества копий
    public void setCopies(int copies) { no usages
        this.copies = copies;
    }
}

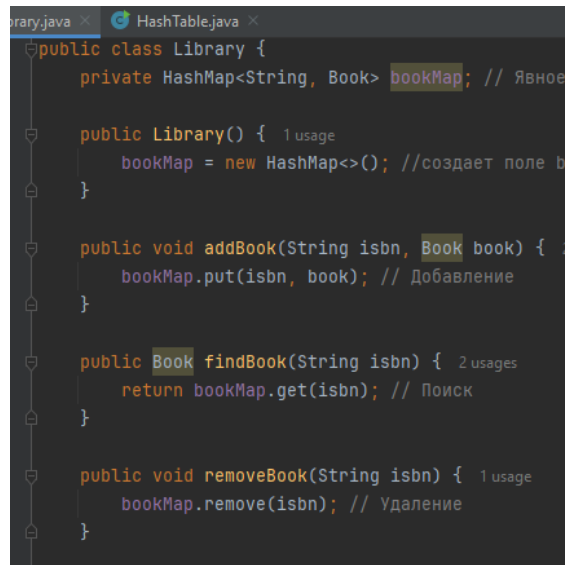
```

Рисунок 6

Класс Book представляет книгу с полями заголовок, автор и количество копий. В классе Library создается HashMap bookMap, где ключом является ISBN книги, а значением объект книги.

- class Book: определяет класс, который представляет книгу с названием, автором и количеством копий.
- private String title;: хранит название книги.
- private String author;: хранит имя автора.
- private int copies;: хранит количество копий книги.
- Book(String title, String author, int copies): конструктор, который инициализирует объект Book с указанными названием, автором и количеством копий.
- getTitle(), getAuthor(), getCopies(): геттеры для получения информации о книге.
- setCopies(int copies): сеттер для изменения количества копий.

Класс Library:



```
library.java x HashTable.java x
public class Library {
    private HashMap<String, Book> bookMap; // Явное

    public Library() { 1 usage
        bookMap = new HashMap<>(); //создает поле bo
    }

    public void addBook(String isbn, Book book) { 2
        bookMap.put(isbn, book); // Добавление
    }

    public Book findBook(String isbn) { 2 usages
        return bookMap.get(isbn); // Поиск
    }

    public void removeBook(String isbn) { 1 usage
        bookMap.remove(isbn); // Удаление
    }
}
```

Рисунок 7

- `public class Library`: определяет класс Library.
- `private HashMap<String, Book> bookMap`:: создает поле bookMap типа HashMap, которое будет хранить книги с ключом ISBN.
- `Library()`: конструктор, который инициализирует bookMap пустым HashMap.

Методы Library:

- `public void addBook(String isbn, Book book)`: добавляет книгу в библиотеку по ее ISBN.
- `public Book findBook(String isbn)`: ищет книгу по ее ISBN и возвращает ее объект, если она найдена, иначе возвращает null.
- `public void removeBook(String isbn)`: удаляет книгу из библиотеки по ее ISBN.

main() метод:

```

public static void main(String[] args) {
    Library library = new Library(); //создает объект Library

    // Добавление книг
    library.addBook( isbn: "978-0143034231", new Book( title: "1984", author: "George Orwell", copies: 5));
    library.addBook( isbn: "978-0141439501", new Book( title: "Pride and Prejudice", author: "Jane Austen", copies: 3));

    // Поиск книги
    Book foundBook = library.findBook( isbn: "978-0143034231");
    if (foundBook != null) { //проверяет, найдена ли книга.
        System.out.println("Найдена книга: " + foundBook.getTitle() + " by " + foundBook.getAuthor());
    } else {
        System.out.println("Книга не найдена.");
    }
}

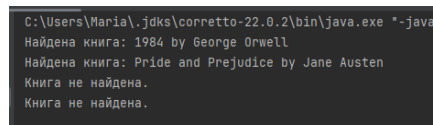
```

Рисунок 8

- `public static void main(String[] args):` точка входа в программу.
- `Library library = new Library();`: создает объект `Library`.
- `library.addBook("978-0143034231", new Book("1984", "George Orwell", 5));`: добавляет книгу “1984” в библиотеку.
- `library.addBook("978-0141439501", new Book("Pride and Prejudice", "Jane Austen", 5));`: добавляет книгу “Pride and Prejudice” в библиотеку.
- `Book foundBook = library.findBook("978-0143034231");`: ищет книгу по ISBN и сохраняет ее в переменную `foundBook`.
- `Book foundBook2 = library.findBook("978-0141439501");`: ищет книгу по ISBN и сохраняет ее в переменную `foundBook`.
- `if (foundBook != null) { ... }`: проверяет, найдена ли книга. Если да, то выводит информацию о ней, иначе выводит сообщение о том, что книга не найдена.

- `library.removeBook("978-0141439501");`: удаляет книгу по ISBN.
- `library.removeBook("978-0141439501");`: удаляет книгу по ISBN.
- `foundBook = library.findBook("978-0141439501");`: ищет удаленную книгу.
- `foundBook2 = library.findBook("978-0141439501");`: ищет удаленную книгу.
- `if (foundBook != null) { ... }`: проверяет, найдена ли книга. Если да, то выводит сообщение о том, что книга найдена, иначе выводит сообщение о том, что книга не найдена.

Вывод в консоли:



```
C:\Users\Maria\.jdk\corretto-22.0.2\bin\java.exe "-java
Найдена книга: 1984 by George Orwell
Найдена книга: Pride and Prejudice by Jane Austen
Книга не найдена.
Книга не найдена.
```

Рисунок 9